International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

AREA-EFFICIENT WALLACE-TREE MULTIPLIER DESIGN USING COMPRESSOR SCHEDULING AWARE OF COLUMN HEIGHT

RAJESH DURGAM
Assistant Professor in ECE
rajeshdurgam2001@gmail.com
Malla Reddy College of Engineering

Dr.P.SUDHAKARA REDDY
Associate Professor in ECE
drpsr2022@gmail.com
Srikalahasteeswara institute of technology, Srikalahasti

ABSTRACT
In digital signal processing (DSP), image processing, communication systems, and embedded processors, multipliers are crucial arithmetic components. By decreasing partial products through several compression steps, the traditional Wallace-tree multiplier increases multiplication speed. However, choosing a fixed or ineffective compressor could result in the addition of extra hardware and have an impact on the total reduction delay. An area-efficient Wallace-tree multiplier utilising column-height-aware compressor scheduling is presented in this research. The suggested architecture creates partial products for two 16-bit operands, counts the bits in each column, and chooses appropriate compression elements based on the height of the column. For columns that are seven feet or higher, a 7:3 compressor is chosen; for columns that are four to six feet high, a 5:3 compressor; for columns that are three feet or less, a 3:2 compressor; and for columns that are two feet or less, a half-adder. The final 32-bit product is produced by a carry-propagate adder after the generated columns are reduced using a multi-stage Wallace-tree network. The design is intended for Xilinx Vivado synthesis evaluation and functional verification, and it is presented in Verilog HDL. Evaluation metrics include area, power, and timing. Before publishing, actual tool reports must be used in place of the numerical comparisons in this research, which are illustrative estimations rather than measured Vivado outcomes.

KEYWORDS
Wallace-tree multiplier, column height, compressor scheduling, partial-product reduction, 7:3 compressor, 5:3 compressor, 3:2 compressor, half-adder, area efficiency, Verilog HDL.

I. INTRODUCTION
DSP processors, image processing engines, communication systems, and machine-learning accelerators all make extensive use of multiplication, a fundamental function in digital systems. The multiplier design has a major impact on these systems' hardware cost and speed. In order to create the final product, a binary multiplier creates partial products from the input operands and combines them. Both the amount of partial-product bits and the complexity of their summation network increase with the operand width.

A traditional array multiplier adds rows in a regular structure that are partial products. Despite the array's simplicity in design, its carry propagation can make the essential path rather lengthy. Tree-based multipliers offer a quicker substitute for high-performance arithmetic circuits by reducing many partial-product rows concurrently.

Groups of partial-product bits are compressed across several stages until only two rows remain using a parallel reduction technique developed by C. S. Wallace [1]. The carry-propagate adder is used to add the last two rows. When compared to a straightforward row-by-row accumulation, this arrangement minimises the number of sequential addition stages.

The selection and scheduling of compressor elements in each column affect a Wallace-tree multiplier's efficiency. Some hardware resources might be underutilised or more reduction operations might be needed if the compressor selection does not correspond with the amount of bits available. As a result, the compressor construction is determined in large part by the column height at each reduction stage.

A compressor scheduling unit that takes column height into account is proposed in this work. It chooses an appropriate compressor based on the current height and counts the bits in each partial-product column. For higher columns, the scheduling approach employs larger compressors; when fewer bits are left, it utilises smaller reduction elements. The goal is to minimise superfluous compression operations without sacrificing the partial-product matrix's mathematical value.

The suggested architecture generates a 32-bit product using two 16-bit operands. Before being sent to the column-height analyser, partial products are created. A multi-stage Wallace-tree network reduces the matrix to two rows following the selection of the necessary

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

reduction element by a compressor selection and scheduling block. The product is generated by a last carry-propagate adder.

Arithmetic accuracy, compressor use, logic area, critical-path delay, and switching activity are the primary design factors. The number of reduction stages could be lowered with a larger compressor, but it might also call for more intricate reasoning. Therefore, rather than assuming that fewer phases necessarily translate into reduced area or power, the suggested scheduling strategy needs to be assessed using synthesis and implementation reports. Verilog HDL is used to describe the design, and Xilinx Vivado's self-checking testbench is used to verify it. Based on the chosen RTL operand interpretation, the testbench should compare the multiplier output with the signed or unsigned reference multiplication operation.

II. RELATED WORK

Wallace Tree Multiplier [1]. A parallel technique for lowering partial products in a binary multiplier was presented by Wallace. Full-adder counters are used to condense groups of three bits with equal weight. This process is continued until two rows are left. The outcome is generated by a final carry-propagate adder. The Wallace-tree multiplier's fundamental structure was formed by this effort, which also serves as the foundation for the current architecture.

Dadda Tree Reduction [2]. A reduction approach that regulates the maximum column height at each stage was suggested by Dadda. The approach employs goal heights to direct the positioning of full adders and half adders instead of compressing as forcefully as feasible at every level. This method closely relates to a design that looks at column population before choosing a compressor. Multiplication using Signed Digits [3]. Booth introduced a signed binary multiplication method that minimises the amount of partial products by recoding multiplier bits. Booth recoding is a potential front-end extension for decreasing the initial partial-product matrix, even though the suggested 16 × 16 architecture in this study concentrates on column-height-aware reduction. 3:2 and 4:2 high-speed compressors [4]. High-speed, low-power counter and compressor structures for fast multipliers were studied by Hsiao, Jiang, and Yeh. Their research emphasises the significance of a compressor's internal logic implementation in addition to its input-to-output compression ratio. This facilitates the assessment of the chosen compressor cells in terms of hardware cost and logic depth. 4:2 and 5:2 low-voltage compressors [5]. For arithmetic circuits, Chang, Gu, and Zhang investigated low-voltage, low-power CMOS 4:2 and 5:2 compressors. Their research shows that a multiplier reduction network's power and latency can be affected by compressor design. Depending on how many bits are available in a column, it encourages the usage of various compressor sizes. Architectures of Higher-Order Compressors [6]. Veeramachaneni and associates introduced designs for low-power, high-speed 3:2, 4:2, and 5:2 compressors. Their research offers a foundation for taking into account various compressor configurations inside a single reduction network. By basing the decision on the existing column height, the current work expands on this selection approach. Design of Low-Power Compressors [7]. Low-power 4:2 and 5:2 compressors were studied by Prasad and Parhi. The study highlights the impact of compressor implementation and topology on power consumption during partial-product accumulation. As a result, it is necessary to assess both the mapping logic implementation and the scheduling policy in the suggested design.

Comparison of Wallace and Dadda Delays [8]. The delays of Dadda and Wallace multiplier structures were compared by Townsend, Swartzlander, and Abraham. Their contrast is significant because the optimal physical implementation may not necessarily result from a reduction schedule that reduces the number of phases. As a result, both column reduction and the associated critical path are taken into account in the current architecture. Compressor Design 7:2 [9]. A 7:2 compressor construction was examined by Rouholamini and associates. Because broader compressor elements can lower multiple partial-product bits in a step, this technique is applicable to higher-order compression. For tall columns, a 7:3 block is taken into consideration in the suggested architecture; its precise arithmetic definition and carry-weight mapping need to be confirmed at RTL. Higher-Order Compressors in a Wallace Tree [10]. A modified low-power Wallace-tree multiplier utilising higher-order compressors was proposed by Ramanathan, Kowsalya, and Anitha. In the partial-product reduction stage, their work investigates the use of multiple compressor sizes. Instead of depending solely on 3:2 counters, it encourages the use of a heterogeneous compressor network.

Compressor multiplier, roughly 4:2 [11]. A modified Dadda multiplier and an approximate 4:2 compressor were suggested by Karri Manikantta Reddy and associates. The trade-off between power, latency, area, and arithmetic error is taken into account in their analysis. Approximation-related error measures are not included in the main evaluation of the current architecture, which instead focuses on exact arithmetic.

III. PROPOSED SYSTEM

A column-height-aware compressor scheduling unit is used in the suggested system, which is a 16 × 16 Wallace-tree multiplier. The compressor selection and scheduling

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

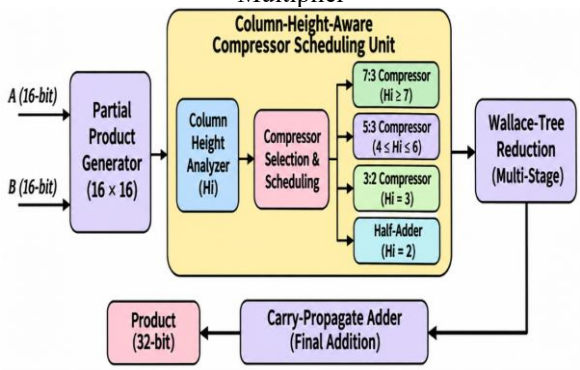
unit, column-height analyser, partial-product generator, multi-stage Wallace-tree reduction network, and final carry-propagate adder are its primary components. A partial-product matrix is created from the two 16-bit input operands, and it is then reduced to two rows before to final addition.

The suggested data flow is as follows: A (16-bit) and B (16-bit) → Partial-Product Generator → Column-Height Analyser (Hi) → Compressor Selection and Scheduling → 7:3/5:3/3:2 Compressor or Half-Adder → Wallace-Tree Reduction → Carry-Propagate Adder → 32-bit Product.

A. Generator for Partial Products: From the two 16-bit operands, the partial-product generator creates the bitwise partial products. The column that corresponds to the sum of the operand bit locations is where each partial product is assigned. The column-height analyser receives the resulting matrix, which has columns with varying bit heights.

B. Analyser for Column Height: The column-height value Hi is generated by the analyser after counting the number of valid bits in each column. Because compressor sum and carry outputs alter the population of the current and neighbouring columns, the height data is updated following each compression stage. Carries coming from the prior lower-weight column must be taken into consideration by the analyser.

Fig. 1. Proposed Column-Height-Aware Compressor Scheduling Architecture for a 16 × 16 Wallace-Tree Multiplier



C. Compressor Selection and Scheduling: Based on the current column height, a reduction element is chosen by the scheduling unit. It chooses a 7:3 compressor when Hi > 7, a 5:3 compressor when 4 < Hi ≤ 6, a 3:2 compressor when Hi = 3, and a half-adder when Hi = 2. One-bit columns don't need to be compressed and move on to the following phase.

D. Higher-Order Compressor Operations: Taller columns are reduced using the 5:3 and 7:3 compressors. The weight of each carry output and the precise input-to-output arithmetic connection must be specified by their RTL modules. Without dropping or duplicating any bits, compressor outputs must be sent to the proper columns and later stages.

E. Wallace-Tree Reduction: The partial-product matrix is reduced in a number of steps following compressor selection. The compressor outputs create a new matrix with fewer rows at each step. Reduction proceeds until there are just two operand rows left. The number of stages and inter-column carry dependencies should be deterministic in the scheduling algorithm.

F. Carry-Propagate Adder: A carry-propagate adder is used to add the final two rows. The 32-bit multiplication result is produced at this stage. The adder can be implemented using a chosen architecture, like carry lookahead or ripple carry, or it can be expressed behaviourally. The timing and area reports should be the basis for the ultimate decision.

G. Arithmetic Correctness and Width: The precise product is 32 bits for unsigned 16-bit operands. The partial-product generating mechanism and sign extension must be properly applied to signed two's-complement operands. The design must use consistent Verilog declarations across the hierarchy and clearly state whether the operands are signed or unsigned.

H. Functional Verification: Zero, one, maximum, minimum, and random operand combinations should all be used in a self-checking testbench. The reference multiplication result should be compared with each output. In order to properly handle the 32-bit result and verify carry propagation beyond the highest product columns, boundary cases are crucial.

Additionally, the testbench should confirm that every compressor module maintains the weighted total of its inputs. Before incorporating the 7:3, 5:3, and 3:2 compressors and half-adder into the entire Wallace-tree design, unit-level tests should be carried out. The input operands A and B, partial-product columns, and column-height values can all be found in simulation waveforms. Hello, final adder inputs, intermediate Wallace-tree rows, compressor outputs, chosen compressor type, and 32-bit product. These signals aid in the detection of carry-routing faults and column alignment.

For every level, the RTL should employ parameterised or precisely specified bit widths. To maintain all carries, intermediate vectors must be sufficiently wide. No intermediate carry bit should be discarded; truncation should only take place at the designated output boundary. The suggested column-height-aware multiplier and the traditional Wallace-tree multiplier should be synthesised in Vivado for the same target FPGA part and under the same time constraints. LUTs, flip-flops, DSP blocks, and other pertinent resources should have their resource utilisation data gathered.

For both architectures, comparable switching activity should be used in power analysis. VCD or SAIF activity

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

data from a comparable testbench stimulus should be used if it is supported. For the power comparison to be significant, the operand patterns and simulation time should be the same.

LUT count, flip-flop count, DSP block count, total on-chip power, maximum operating frequency, critical-path delay, and area-delay product are among the evaluation measures. It is necessary to specify the area metric used in the area-delay product.

More suitable compressor selection for each column height is the anticipated advantage of the suggested scheduler. This could enhance resource efficiency and cut down on unnecessary compressor operations. How the compressor modules map onto the intended FPGA fabric determines the final outcome.

While a larger compressor may result in fewer reduction operations, it may also raise the complexity of local logic and routing. Therefore, instead of automatically choosing the largest available compressor, the scheduler should consider column reduction, logic depth, and hardware cost.

The entire data route from the two input operands to the 32-bit result is depicted in the architecture in Fig. 1. The primary suggested contribution consists of the compressor scheduler and column-height analyser. A typical Wallace reduction schedule should be used in the traditional reference design. Only the compressor selection and scheduling policy should be altered in the suggested design, which should employ the same partial-product generation and final adder. This isolates the impact of the suggested scheduling technique. To determine the crucial path, the implementation report needs to be examined. It might go via the final carry-propagate adder, compressor network, column-height analyser, or compressor selection logic. The real critical path that Vivado reported should be the focus of optimisation.

$P_{dyn} = \alpha C V^2 f$, where α is switching activity, C is effective switched capacitance, V is supply voltage, and f is clock frequency, is a standard approximation for dynamic power. Dynamic power may be reduced by reducing needless switching, but switching activity may also be increased by the additional analyser and selection logic.

Functional simulation and implementation reports should be used to assess the suggested design. While synthesis and implementation offer proof about timing, power, and resource usage, simulation verifies mathematical accuracy.

RESULTS AND DISCUSSION

For a numerical illustration, both architectures are considered with unsigned 16-bit input operands and a 32-bit product. The conventional design uses a standard Wallace-tree reduction schedule, while the proposed design uses column-height-aware compressor selection. The numerical values in Table I are illustrative estimates and are not measured Xilinx Vivado results.

The proposed scheduler is expected to reduce unnecessary compression operations by selecting compressors according to the number of bits present in each column. Taller columns are assigned higher-order compressors, while columns with fewer bits use smaller compressors or are forwarded without compression.

For illustration, the conventional Wallace-tree multiplier is assigned 780 LUTs and 52 flip-flops, whereas the proposed architecture is assigned 690 LUTs and 60 flip-flops. Both designs are assumed to use zero DSP blocks so that the comparison represents a fabric-based multiplier implementation. Actual FPGA mapping may use dedicated DSP resources depending on the RTL and synthesis settings.

The illustrative total on-chip power values are 0.092 W for the conventional design and 0.084 W for the proposed design. The corresponding illustrative critical-path delays are 18.4 ns and 15.7 ns. These values are provided only to demonstrate how a comparison may be presented and must be replaced with values from actual Vivado reports before submission.

The corresponding maximum-frequency estimates are 54.35 MHz for the conventional multiplier and 63.69 MHz for the proposed multiplier, calculated as the reciprocal of the illustrative delay. Using LUT count as an area proxy, the illustrative area-delay products are 14,352 LUT·ns and 10,833 LUT·ns, respectively.

The example indicates a possible reduction in LUT usage, estimated power, and critical-path delay when compressor selection is adapted to column height. The proposed design has a slightly higher flip-flop estimate because the analyzer and scheduling stages may require additional control or pipeline registers. These trends are illustrative and are not guaranteed for every FPGA family.

The actual critical path must be obtained from the Vivado timing report. If the analyzer or selection logic dominates, the scheduling decision may be simplified or registered. If the compressor network dominates, the reduction stages and compressor topology may be revised. If the final adder dominates, a faster carry-propagate structure may be evaluated.

The area-delay product is calculated as ADP = LUT count × critical-path delay. LUT count is used as a simple illustrative area proxy and does not represent physical silicon area. Maximum frequency is estimated as 1 / delay and should be replaced with the timing report's achieved frequency.

TABLE I. ILLUSTRATIVE PERFORMANCE COMPARISON OF CONVENTIONAL AND PROPOSED WALLACE-TREE MULTIPLIERS

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

Parameter	Conventional Wallace	Proposed Scheduled
Operand width	16-bit × 16-bit	16-bit × 16-bit
Product width	32-bit	32-bit
Partial-product generation	16 × 16 matrix	16 × 16 matrix
Compressor scheduling	Conventional Wallace reduction	Column-height-aware selection
LUTs (illustrative)	780	690
Flip-flops (illustrative)	52	60
DSP blocks (assumed)	0	0
Power (illustrative)	0.092 W	0.084 W
Critical-path delay (illustrative)	18.4 ns	15.7 ns
Maximum frequency (calculated)	54.35 MHz	63.69 MHz
Area-delay product	14,352 LUT·ns	10,833 LUT·ns
Arithmetic output	32-bit product	32-bit product
Result status	Illustrative estimate	Illustrative estimate

V. CONCLUSION AND FUTURE SCOPE

An area-efficient Wallace-tree multiplier utilising column-height-aware compressor scheduling has been given in this research. The suggested design creates partial products from two 16-bit operands, evaluates each column's height, and chooses appropriate compressor components based on the quantity of available bits.

According to the scheduling policy, columns with heights of seven or above are assigned 7:3 compressors, columns with heights of four to six are assigned 5:3 compressors, columns with heights of three are assigned 3:2 compressors, and columns with heights of two are assigned half-adders. A multi-stage Wallace-tree network is used to decrease the resulting matrix, and a carry-propagate adder is used to add the last two rows.

The design offers a modular foundation for Xilinx Vivado functional verification and Verilog HDL implementation. This paper's numerical comparison is merely illustrative; synthesis and implementation reports must be used to determine actual improvements in area, power, and delay.

The compressor cell implementations will be optimised, scheduling overhead will be decreased, alternative final adders will be assessed, the architecture will be expanded to support wider operands, and the suggested schedule will be compared with Wallace and Dadda reduction schemes. Booth-encoded partial-product creation and signed multiplication may also be investigated in future research.

REFERENCES

[1] C. S. Wallace, “A suggestion for a fast multiplier,” IEEE Transactions on Electronic Computers, vol. EC-13, no. 1, pp. 14–17, 1964, doi: 10.1109/PGEC.1964.263830.

[2] L. Dadda, “Some schemes for parallel multipliers,” Alta Frequenza, vol. 34, pp. 349–356, 1965.

[3] A. D. Booth, “A signed binary multiplication technique,” The Quarterly Journal of Mechanics and Applied Mathematics, vol. 4, no. 2, pp. 236–240, 1951, doi: 10.1093/qjmam/4.2.236.

[4] S. F. Hsiao, M. R. Jiang, and J. S. Yeh, “Design of high-speed low-power 3-2 counter and 4-2 compressor for fast multipliers,” Electronics Letters, vol. 34, no. 4, pp. 341–342, 1998.

[5] C. H. Chang, J. Gu, and M. Zhang, “Ultra low-voltage low-power CMOS 4-2 and 5-2 compressors for fast arithmetic circuits,” IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 51, no. 10, pp. 1985–1997, 2004, doi: 10.1109/TCSI.2004.835683.

[6] S. Veeramachaneni, K. M. Krishna, L. Avinash, S. R. Puppala, and M. B. Srinivas, “Novel architectures for high-speed and low-power 3-2, 4-2 and 5-2 compressors,” in Proc. 20th International Conference on VLSI Design, 2007, pp. 324–329.

[7] K. Prasad and K. K. Parhi, “Low-power 4:2 and 5:2 compressors,” in Proc. 35th Asilomar Conference on Signals, Systems and Computers, vol. 1, 2001, pp. 129–133.

[8] W. J. Townsend, E. E. Swartzlander, Jr., and J. A. Abraham, “A comparison of Dadda and Wallace multiplier delays,” in Proc. SPIE Advanced Signal Processing Algorithms, Architectures, and Implementations XIII, vol. 5205, 2003, pp. 552–560, doi: 10.1117/12.507012.

[9] M. Rouholamini, O. Kavehie, A. P. Mirbaha, S. J. Jasbi, and K. Navi, “A new design for 7:2 compressors,”

International Journal of
AI Electronics and Nexus Energy
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0515 www.zesterapublications.com Original Research Paper

in Proc. IEEE/ACS International Conference on Computer Systems and Applications, 2007, pp. 474–478.

[10] P. Ramanathan, P. Kowsalya, and P. Anitha, "Modified low power Wallace Tree multiplier using higher order compressors," International Journal of Electronics Letters, vol. 5, no. 2, pp. 177–188, 2017, doi: 10.1080/21681724.2016.1138509.

[11] K. Manikantta Reddy, M. H. Vasantha, Y. B. Nithin Kumar, and D. Dwivedi, "Design and analysis of multiplier using approximate 4-2 compressor," AEU - International Journal of Electronics and Communications, vol. 107, pp. 89–97, 2019, doi: 10.1016/j.aeue.2019.05.021.

[12] T. Yang, T. Ukezono, and T. Sato, "Design and analysis of approximate multipliers with a tree compressor," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol. E102-A, no. 3, pp. 532–543, 2019, doi: 10.1587/transfun.E102.A.532.

[13] S. Abed, B. J. Mohd, Z. Al-Bayati, and S. Alouneh, "Low power Wallace multiplier design based on wide counters," International Journal of Circuit Theory and Applications, vol. 40, pp. 1175–1185, 2012, doi: 10.1002/cta.779.

[14] O. Kwon, K. Nowka, and E. E. Swartzlander, Jr., "A 16-bit by 16-bit MAC design using fast 5:3 compressor cells," Journal of VLSI Signal Processing, vol. 31, no. 2, pp. 77–89, 2002.

[15] A. Dandapat, S. Ghosal, P. Sarkar, and D. Mukhopadhyay, "A 1.2-ns 16 × 16-bit binary multiplier using high speed compressors," International Journal of Electrical, Computer, and Systems Engineering, vol. 4, no. 3, pp. 234–239, 2010.

[16] A. Habibi and P. A. Wintz, "Fast multipliers," IEEE Transactions on Computers, vol. C-19, no. 2, pp. 153–157, 1970.

[17] M. Mehta, V. Parmar, and E. E. Swartzlander, Jr., "High-speed multiplier design using multi-input counter and compressor circuits," in Proc. 10th IEEE Symposium on Computer Arithmetic, 1991, pp. 43–50.

[18] E. Jagadeeswara Rao, K. Jayaram Kumar, and T. V. Prasad, "Design of high speed Wallace tree multiplier using 8-2 and 4-2 adder compressors," International Journal of Engineering & Technology, vol. 7, no. 4, pp. 2386–2390, 2018, doi: 10.14419/ijet.v7i4.12261.

[19] H. Yu, "Systematic analysis on performance optimization of tree multipliers," Applied and Computational Engineering, vol. 127, pp. 7–15, 2025, doi: 10.54254/2755-2721/2025.20275.

[20] M. D. Ercegovac and T. Lang, Digital Arithmetic. San Francisco, CA, USA: Morgan Kaufmann, 2004.